Data Compression
Lecture Notes

Dr. Faisal Aslam

Lecture 1: Introduction to Data Compression

Learning Objectives

By the end of this lecture, students will be able to:

- Define data compression and explain its practical importance with real-world examples
- Differentiate between lossless and lossy compression with concrete applications
- Calculate and interpret basic compression metrics (compression ratio, bit-rate, savings)
- Explain the concepts of information, redundancy, and entropy with computational examples
- Identify major application domains and their specific compression requirements
- Understand the fundamental limits of compression from information theory

0.1 Introduction and Motivation: Why Compress Data?

Data compression is the process of encoding information using fewer bits than the original representation. Every day, we encounter compression without realizing it: from streaming videos to sending emails, from saving photos to downloading software updates.

Definition

Data Compression: The process of reducing the number of bits needed to represent information, while either:

- **Lossless**: Preserving all original information exactly
- **Lossy**: Accepting some controlled loss of information for higher compression

0.1.1 Real-World Motivation: A Concrete Example

Consider a typical smartphone photo: 12 megapixels, 24-bit color (8 bits per RGB channel). Uncompressed size:

$$12,000,000 \text{ pixels} \times 24 \text{ bits/pixel} = 288,000,000 \text{ bits} = 36 \text{ MB}$$

But your phone stores it as a 3 MB JPEG file. That's a 12:1 compression ratio! Without compression:

- Your 128 GB phone could store only 3,500 photos instead of 40,000
- Uploading to social media would take 12 times longer
- Cloud storage costs would be 12 times higher

0.1.2 The Economics of Compression

Example

Cloud Storage Example: A major cloud provider charges \$0.023 per GB per month. For 1 PB (petabyte = 1000 TB) of data:

- Uncompressed: 1 PB = 1,000,000 GB gives \$23,000/month
- With 4:1 compression: 250,000 GB gives \$5,750/month
- Annual savings: (\$23,000 - \$5,750) $\times$ 12 = \$207,000/year

This doesn't even consider bandwidth costs, which are typically charged per GB transferred!

0.2 Lossless vs. Lossy Compression: A Detailed Comparison

0.2.1 Lossless Compression: Perfect Reconstruction

How it works: Exploits statistical redundancy and patterns without losing information.
Key techniques:

1. **Entropy coding**: Assign shorter codes to frequent symbols (Huffman, Arithmetic)
2. **Dictionary methods**: Replace repeated patterns with references (LZ77, LZ78)
3. **Predictive coding**: Encode differences from predictions rather than raw values

Example

Text Compression Example: The word "compression" appears 100 times in a document.

- Uncompressed: "compression" = 11 characters $\times$ 8 bits = 88 bits $\times$ 100 = 8,800 bits
- Compressed: Assign code "01" (2 bits) for "compression" $\rightarrow$ 2 bits $\times$ 100 = 200 bits
- Plus dictionary entry: "compression" = 88 bits (stored once)
- Total: 200 + 88 = 288 bits vs 8,800 bits $\rightarrow$ 30:1 compression!

This is essentially how LZW (used in GIF, ZIP) works.

0.2.2 Lossy Compression: Intelligent Approximation

How it works: Removes information that is:

- Imperceptible to humans (psychovisual/psychoacoustic models)
- Less important for the intended use
- Redundant beyond a certain quality threshold

Example

JPEG Image Compression - Step by Step:

1. **Color Space Conversion**: RGB to YCbCr (separates luminance from color)
2. **Chrominance Downsampling**: Reduce color resolution (4:2:0) - humans are less sensitive to color details
3. **Discrete Cosine Transform (DCT)**: Convert 8×8 pixel blocks to frequency domain
4. **Quantization**: Divide frequency coefficients by quantization matrix - small high-frequency coefficients become zero
5. **Entropy Coding**: Huffman code the results

   Result: Typical 10:1 to 20:1 compression with minimal visible artifacts

Factor	Choose Lossless When	Choose Lossy When
Fidelity Requirement	Exact reconstruction is critical (code, financial data, legal documents)	Some quality loss is acceptable (media streaming, web images)
Data Type	Discrete data with exact values (text, databases, executables)	Continuous data with perceptual limits (images, audio, video)
Compression Ratio Needed	Moderate ratios suffice (2:1 to 10:1)	High ratios needed (10:1 to 200:1+)
Processing Requirements	Fast decompression needed, encode speed less critical	Real-time encoding/decoding needed (streaming, videoconferencing)
Regulatory Constraints	Legal/medical requirements mandate exact copies	No regulatory constraints on quality

Table 1: Decision Factors for Lossless vs. Lossy Compression

0.2.3 When to Use Which? Decision Factors

0.3 Performance Metrics: Beyond Simple Ratios

0.3.1 Compression Ratio and Savings

$$\text{Compression Ratio} = \frac{\text{Original Size}}{\text{Compressed Size}}$$
$$\text{Savings2} = \left(1 - \frac{\text{Compressed Size}}{\text{Original Size}}\right) \times 100\%$$

Example

Comparing Different Compression Scenarios:

Scenario	**Original**	**Compressed**	**Ratio**	**Savings**
Text document (ZIP)	1.5 MB	450 KB	3.33:1	70%
CD Audio (FLAC lossless)	700 MB	350 MB	2:1	50%
Same Audio (MP3 128kbps)	700 MB	112 MB	6.25:1	84%
4K Video (H.265)	100 GB	2 GB	50:1	98%
DNA sequence (specialized)	3 GB	300 MB	10:1	90%

0.3.2 Bit-rate: The Quality Control Knob

For lossy compression, bit-rate determines quality:

$$\text{Bit-rate} = \frac{\text{Compressed Size in bits}}{\text{Duration (seconds)}} \quad \text{or} \quad \frac{\text{Compressed Size in bits}}{\text{Number of samples}}$$

Example

Audio Quality at Different Bit-rates:

- **32 kbps**: Telephone quality, speech only
- **96 kbps**: FM radio quality
- **128 kbps**: ”Good enough” for most listeners
- **192 kbps**: Near CD quality for most people
- **320 kbps**: Essentially transparent (FLAC: 900 kbps)

  Storage impact: A 60-minute album:

  – At 128 kbps: 60 MB
  – At 320 kbps: 144 MB
  – FLAC lossless: 400 MB
  – Uncompressed CD: 700 MB

0.3.3 Time and Space Trade-offs

Compression involves multiple dimensions:

$$\text{Space-Time Trade-off} = \frac{\text{Compression Ratio}}{\text{Encoding Time} \times \text{Decoding Time}}$$

Example

Real-world Compressor Comparison:

Algorithm	**Ratio (text)**	**Encode Speed**	**Decode Speed**	**Memory**
gzip (-6)	3.2:1	100 MB/s	400 MB/s	10 MB
bzip2 (-6)	3.8:1	20 MB/s	50 MB/s	50 MB
LZ4	2.5:1	500 MB/s	2000 MB/s	1 MB
Zstd (-3)	3.0:1	300 MB/s	800 MB/s	5 MB
xz (-6)	4.2:1	10 MB/s	80 MB/s	100 MB

Approximate performance on typical text data (higher is better)

0.4 Information and Redundancy: The Core Concepts

0.4.1 Information: Quantifying Surprise

Claude Shannon’s revolutionary insight: Information is inversely related to probability.

Definition

Information Content of an event with probability p:

$$I(p) = -\log_2 p \quad \text{bits}$$

Example

Daily Weather Forecast - Information Content:

- Sunny in Phoenix (probability 0.9): $I = -\log_2 0.9 \approx 0.15$ bits
- Snow in Phoenix (probability 0.001): $I = -\log_2 0.001 \approx 9.97$ bits
- Rain in Seattle (probability 0.3): $I = -\log_2 0.3 \approx 1.74$ bits

  Interpretation: Rare events carry more information! Snow in Phoenix tells you much more about the weather pattern than yet another sunny day.

0.4.2 Redundancy: The Enemy of Compression

Redundancy comes in several forms:

1. **Spatial Redundancy**: Neighboring pixels are correlated

   Example

   In a blue sky photo, most pixels are similar shades of blue. Instead of storing each pixel independently:

   - Naive: RGB values for each of 1 million pixels
   - Smart: "The next 1000 pixels are color (135, 206, 235)" - Run-length encoding
   - Even smarter: Predict each pixel from its neighbors, encode only differences

2. **Statistical Redundancy**: Uneven symbol frequencies

Example

English text letter frequencies:

Letter	Frequency	Letter	Frequency
E	12.7%	Z	0.07%
T	9.1%	Q	0.10%
A	8.2%	J	0.15%

Inefficient: Fixed 5 bits per letter (32 possible) **Efficient**: Huffman coding: E = 3 bits, Z = 9 bits Average bits per letter drops from 5 to 4.1

3. **Knowledge Redundancy**: Information known to both encoder and decoder

Example

Medical Imaging: Both sides know the image represents a chest X-ray:

- Don't need to encode that lungs should be in certain positions
- Can use anatomical models to predict and encode differences
- Can focus bits on diagnostically important regions

4. **Perceptual Redundancy**: Information humans can't perceive

Example

Audio Compression (MP3):

- **Frequency masking**: A loud sound at 1 kHz makes nearby frequencies (950-1050 Hz) inaudible
- **Temporal masking**: A loud sound makes preceding/following quiet sounds inaudible
- **Result**: Can discard 90% of audio data without audible difference

0.5 Entropy: The Fundamental Limit

0.5.1 Calculating Entropy: Step by Step

Entropy is the average information content per symbol:

Definition

Entropy of a discrete source with symbols $s_1, s_2, \ldots, s_n$ having probabilities $p_1, p_2, \ldots, p_n$:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i \quad \text{bits per symbol}$$

Example

Binary Source Example - Detailed Calculation: Consider a biased coin: P(Heads) = 0.8, P(Tails) = 0.2

1. Information content of Heads: $I_H = -\log_2 0.8 \approx 0.3219$ bits
2. Information content of Tails: $I_T = -\log_2 0.2 \approx 2.3219$ bits
3. Entropy: $H = 0.8 \times 0.3219 + 0.2 \times 2.3219 = 0.7219$ bits

Interpretation:

- On average, each coin flip gives us 0.72 bits of information
- We need at least 0.72 bits per flip to encode the sequence
- If coins were fair (P=0.5), $H = 1.0$ bit - maximum uncertainty
- If always heads (P=1.0), $H = 0$ bits - no information

0.5.2 Entropy of English Text

Example

Calculating English Letter Entropy:

Letter	Probability	$-\log_2 p$	Contribution
E	0.127	2.98	0.378
T	0.091	3.46	0.315
A	0.082	3.61	0.296
...	...	...	...
Z	0.0007	10.48	0.007
Total			**4.18 bits**

What this means:

- **Naive encoding**: 5 bits per letter (32 possibilities)
- **Entropy limit**: 4.18 bits per letter
- **Practical Huffman**: 4.3 bits per letter
- **With word models**: 2.3 bits per letter (exploiting word-level patterns)
- **With context**: 1.5 bits per letter (exploiting grammar, semantics)

0.5.3 The Entropy Theorem: Why It Matters

Important

Shannon's Source Coding Theorem (Informal):

- **Lower bound**: No lossless compressor can average fewer than H bits/symbol
- **Upper bound**: You can get arbitrarily close to H bits/symbol
- **Implication**: Entropy is the absolute limit for lossless compression

  Example: For English letters ($H = 4.18$ bits):

  - Impossible: Average ¡ 4.18 bits/letter
  - Possible but wasteful: 8 bits/letter (ASCII)
  - Good: 4.3 bits/letter (Huffman)
  - Approaching limit: 4.19 bits/letter (Arithmetic with context)

0.6 Application Domains: Specialized Requirements

0.6.1 Text and Code Compression

- **Requirements**: Lossless, fast random access, incremental updates
- **Challenges**: Small files, need for searching within compressed data
- **Solutions**: gzip (DEFLATE), LZ4, Zstandard

Example

Git Version Control: Uses zlib (DEFLATE) for compression:

- Stores each file version compressed separately
- Achieves 2:1 to 10:1 compression on source code
- Fast decompression for diffs and merges
- Example: Linux kernel repository: 4 GB uncompressed, 1 GB compressed

0.6.2 Multimedia Compression

- **Requirements**: High compression, perceptual quality, real-time
- **Challenges**: Massive data volumes, human perception constraints
- **Solutions**: JPEG, MP3, H.264/HEVC, AV1

Example

Streaming Service Economics (Netflix/YouTube):

- 1 hour of 4K video: Uncompressed 500 GB
- H.265 compressed: 4 GB (125:1 compression)
- Bandwidth cost: $0.05/GB (typical CDN pricing)
- Uncompressed stream: $25/hour
- Compressed stream: $0.20/hour
- For 100 million hours/day: $20M/day vs $2.5B/day!

0.6.3 Scientific and Medical Data

- **Requirements**: Lossless or controlled loss, reproducibility, standards
- **Challenges**: Huge datasets, precision requirements, regulatory compliance
- **Solutions**: Specialized compressors (SZ, ZFP), format standards (DICOM)

Example

Large Hadron Collider (LHC) Data:

- Generates 1 PB/second (yes, per second!)
- Stores 50 PB/year after filtering
- Uses specialized compression algorithms
- Compression saves \$50M/year in storage costs
- Enables global collaboration (data distributed worldwide)

0.7 The Compression Pipeline: How Compressors Actually Work

Most compressors follow this two-stage process:

Compression Pipeline:

Input Data $\xrightarrow[\text{Analyzes patterns}]{}$ **Modeling Stage** $\xrightarrow[\text{Builds probability model}]{}$

$\longrightarrow$ **Coding Stage** $\xrightarrow[\text{Using entropy coding}]{}$

Converts to bits

$\longrightarrow$ **Compressed Data**

Two-Stage Compression Pipeline:

- **Modeling Stage**: Analyzes data patterns and builds probability model
- **Coding Stage**: Converts symbols to bits using entropy coding (Huffman, Arithmetic, ANS)

Example

Huffman Coding Example - Complete Process:

1. **Modeling**: Count symbol frequencies in "ABRACADABRA"

Symbol	Frequency	Probability
A	5	$5/11 \approx 0.455$
B	2	$2/11 \approx 0.182$
R	2	$2/11 \approx 0.182$
C	1	$1/11 \approx 0.091$
D	1	$1/11 \approx 0.091$

2. **Coding**: Build Huffman tree (simplified):
   - Combine lowest frequencies: C(1) + D(1) = CD(2)
   - Continue combining: CD(2) + B(2) = CDB(4)
   - Combine: CDB(4) + R(2) = CDBR(6)
   - Final: CDBR(6) + A(5) = Root(11)

3. **Code assignment**:

Symbol	Code	Length
A	0	1 bit
R	10	2 bits
B	110	3 bits
C	1110	4 bits
D	1111	4 bits

4. **Compress "ABRACADABRA"**:
   - A(0) B(110) R(10) A(0) C(1110) A(0) D(1111) A(0) B(110) R(10) A(0)
   - Total bits: 1+3+2+1+4+1+4+1+3+2+1 = 23 bits
   - Original: 11 characters × 8 bits = 88 bits
   - Compression: 88 → 23 bits (3.8:1 ratio)
   - Entropy limit: $H \approx 2.04$ bits/char × 11 = 22.5 bits
   - Efficiency: 22.5/23 = 97.8% efficient!

0.8 Important Terminology and Concepts

0.8.1 Key Definitions with Examples

- **Symbol**: The basic unit being compressed

Example

Different domains use different symbols:

  - Text: Characters (bytes)
  - Images: Pixels (RGB triples)
  - Audio: Samples (16-bit integers)
  - Video: Macroblocks (16×16 pixel regions)

- **Alphabet**: Set of all possible symbols

Example

  - English text: 256 possible bytes (ASCII/UTF-8)
  - Binary data: 256 possible byte values
  - DNA sequences: 4 symbols {A, C, G, T}
  - Black-white image: 2 symbols {0=black, 1=white}

- **Prefix Code**: Crucial for instant decoding

Example

Why prefix codes matter:

  - Good: A=0, B=10, C=110, D=111
  - ”010110” decodes unambiguously: A(0) B(10) C(110)
  - Bad: A=0, B=1, C=01 (not prefix-free)
  - ”01” could be AB or C - ambiguous!

0.8.2 The Fundamental Insight

Important

The Core Principle of Compression:

- **Random data cannot be compressed**: Maximum entropy = no redundancy
- **Real-world data is not random**: Contains patterns, structure, predictability
- **Compression finds and exploits these patterns**

  Example - Encryption vs Compression:

  – Encrypted data looks random (high entropy)
  – Compressing encrypted data gives little or no savings
  – Always compress **before** encrypting, not after!
  – Rule: Encrypt $\rightarrow$ High entropy $\rightarrow$ No compression
  – Rule: Compress $\rightarrow$ Lower entropy $\rightarrow$ Then encrypt

0.9 Homework Assignment: Practical Exercises

1. **Compression Calculation**:
   - A 4K video frame is 3840×2160 pixels, 24-bit color. Calculate:
     (a) Uncompressed size in MB
     (b) Size after 10:1 compression
     (c) Size after 50:1 compression
     (d) For a 2-hour movie at 24 fps, calculate total sizes
2. **Entropy Calculation**:
   - Calculate entropy for these sources:
     (a) A die roll (6 equally likely outcomes)
     (b) Weather: Sunny(0.6), Cloudy(0.3), Rainy(0.1)
     (c) Binary source: P(0)=0.99, P(1)=0.01
   - Which is most compressible? Why?
3. **Real-world Analysis**:
   - Take three files from your computer: a .txt document, a .jpg image, and a .zip file

- Record their sizes
- Compress them using gzip at maximum compression
- Calculate compression ratios
- Explain why they compress differently

4. **Huffman Coding Practice**:
   - For the message "MISSISSIPPI":
     (a) Calculate symbol frequencies
     (b) Build Huffman tree
     (c) Assign codes
     (d) Encode the message
     (e) Calculate compression ratio vs 8-bit ASCII
     (f) Compare to entropy limit

5. **Research and Analysis**:
   - Find a current research paper on neural compression
   - Summarize its approach in 200 words
   - Compare its claimed performance to traditional methods
   - Identify one advantage and one limitation

0.10 Looking Ahead: What's Next?

In the next lecture, we will dive deeper into:

- **Shannon's Source Coding Theorem**: Formal statement and proof
- **Kraft-McMillan Inequality**: Mathematical foundation of prefix codes
- **Optimal Code Construction**: How to achieve the entropy limit
- **Practical Implications**: What these theorems mean for real compressors

Important

Key Takeaways from Lecture 1:

1. Compression is economically and practically essential in modern computing
2. Lossless vs lossy involves trade-offs between fidelity and compression ratio
3. Entropy defines the absolute limit for lossless compression
4. Real compressors work by modeling data patterns, then encoding efficiently
5. Different applications require specialized compression approaches

Lecture 2: Shannon's Source Coding and Prefix Codes

Learning Objectives

By the end of this lecture, students will be able to:

- State and interpret Shannon's Source Coding Theorem
- Apply the Kraft-McMillan inequality to validate prefix codes
- Construct prefix codes and understand their properties
- Explain the relationship between entropy and achievable compression
- Calculate code efficiency and redundancy

0.11 Shannon's Source Coding Theorem

Definition

Shannon's Source Coding Theorem (First Theorem): For any discrete memoryless source with entropy $H(X)$, the expected code length L of any uniquely decodable code satisfies:

$$H(X) \leq L < H(X) + 1$$

0.11.1 Interpretation

- **Lower bound**: Cannot compress below entropy on average
- **Upper bound**: Can get within 1 bit of entropy
- **For extended sources**: With n-symbol blocks, can approach $H(X)$ arbitrarily close:
$$H(X) \leq \frac{L_n}{n} < H(X) + \frac{1}{n}$$

Example

For a source with $H(X) = 2.3$ bits/symbol:

- No code can achieve $L < 2.3$ bits/symbol on average
- A good code can achieve $2.3 \leq L < 3.3$ bits/symbol
- By coding in blocks of 10 symbols: $2.3 \leq L/10 < 2.4$ bits/symbol

0.12 Types of Codes

0.12.1 Non-singular Codes

Each source symbol maps to a distinct binary string. Not necessarily uniquely decodable.

0.12.2 Uniquely Decodable Codes

Any finite sequence of codewords corresponds to at most one message.

0.12.3 Prefix Codes (Instantaneous Codes)

No codeword is a prefix of another codeword. This guarantees immediate decodability.

Example

Examples of codes for alphabet $\{a, b, c, d\}$:

Symbol	Non-singular	Uniquely Decodable	Prefix Code
a	0	0	0
b	1	01	10
c	00	011	110
d	11	0111	111

0.13 Kraft-McMillan Inequality

Definition

Kraft Inequality: For any prefix code with codeword lengths $\ell_1, \ell_2, \ldots, \ell_n$:

$$\sum_{i=1}^{n} 2^{-\ell_i} \leq 1$$

McMillan Inequality: The same condition holds for any uniquely decodable code.

0.13.1 Implications

- Provides a necessary condition for prefix codes
- Also sufficient: If lengths satisfy Kraft, a prefix code exists
- Helps find optimal code lengths before constructing the code

Example

Check if these lengths can form a prefix code: $\{2, 2, 3, 3\}$

$$2^{-2} + 2^{-2} + 2^{-3} + 2^{-3} = \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} = 1$$

Yes, a prefix code exists with these lengths.

0.14 Constructing Prefix Codes

0.14.1 Binary Tree Representation

Prefix codes correspond to leaves of a binary tree:

- Each leaf represents a codeword
- Path from root gives the binary code
- Depth equals codeword length

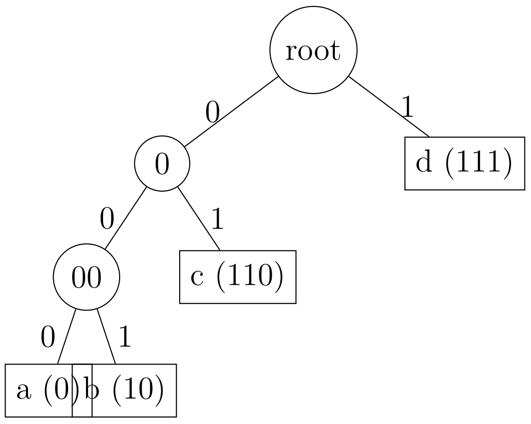

Figure 1: Binary tree representation of prefix code: a=0, b=10, c=110, d=111

0.15 Optimal Code Lengths

For a source with symbol probabilities p_i, optimal codeword lengths satisfy:

$$\ell_i^* = \lceil -\log_2 p_i \rceil$$

These satisfy Kraft inequality and are close to the theoretical minimum.

Example

For symbols with probabilities $\{0.5, 0.25, 0.125, 0.125\}$:

$$\begin{aligned}
\ell_1 &= \lceil -\log_2 0.5 \rceil = \lceil 1 \rceil = 1 \\
\ell_2 &= \lceil -\log_2 0.25 \rceil = \lceil 2 \rceil = 2 \\
\ell_3 &= \lceil -\log_2 0.125 \rceil = \lceil 3 \rceil = 3 \\
\ell_4 &= \lceil -\log_2 0.125 \rceil = \lceil 3 \rceil = 3
\end{aligned}$$

Check Kraft: $2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 0.5 + 0.25 + 0.125 + 0.125 = 1$

0.16 Code Efficiency and Redundancy

Definition

Code Efficiency:

$$\eta = \frac{H(X)}{L} \times 100\%$$

Redundancy:

$$\rho = L - H(X) \quad \text{(bits/symbol)}$$

0.16.1 Upper Bound on Efficiency

From Shannon's theorem:

$$\frac{H(X)}{H(X)+1} \leq \eta \leq 1$$

Example

For a source with $H(X) = 2.3$ bits/symbol and code achieving $L = 2.5$ bits/symbol:

$$\eta = \frac{2.3}{2.5} \times 100\% = 92\%$$

$$\rho = 2.5 - 2.3 = 0.2 \text{ bits/symbol}$$

0.17 Extended Coding (Block Coding)

By coding blocks of n symbols together:

- Achieve average length per symbol: $L_n/n \to H(X)$ as $n \to \infty$
- Redundancy per symbol: $\rho_n < 1/n$
- Trade-off: Larger n requires more memory and complexity

0.18 Proof Sketch of Shannon's Theorem

0.18.1 Lower Bound ($L \geq H(X)$)

1. Let p_i be symbol probabilities, ℓ_i be codeword lengths
2. From information theory: $-\log_2 p_i$ is the ideal length
3. Using Gibbs' inequality or convexity arguments:

$$\sum p_i \ell_i \geq -\sum p_i \log_2 p_i = H(X)$$

0.18.2 Upper Bound ($L < H(X) + 1$)

1. Choose lengths: $\ell_i = \lceil -\log_2 p_i \rceil$
2. Then: $-\log_2 p_i \leq \ell_i < -\log_2 p_i + 1$
3. Multiply by p_i and sum:

$$H(X) \leq \sum p_i \ell_i < H(X) + 1$$

0.19 Applications and Implications

Important

Practical Implications:

- Entropy provides the absolute limit for lossless compression
- Real compressors have additional overhead (model storage, headers)
- For small alphabets, the "+1" term can be significant
- Block coding helps approach the entropy limit

0.20 Limitations and Extensions

0.20.1 Assumptions in Shannon's Theorem

- Discrete memoryless source (symbols independent)
- Known probability distribution
- Noiseless channel
- These assumptions are often violated in practice

0.20.2 Extensions

- **Markov sources**: Consider dependencies between symbols
- **Universal coding**: No prior knowledge of distribution
- **Rate-distortion theory**: Lossy compression extension

0.21 Algorithm: Constructing a Prefix Code from Given Lengths

Algorithm 1 Construct Prefix Code from Lengths

```
Require: Codeword lengths $\ell_1, \ell_2, \ldots, \ell_n$ satisfying Kraft inequality
Ensure: Prefix code with given lengths
1: Sort lengths in non-decreasing order: $\ell_{(1)} \leq \ell_{(2)} \leq \cdots \leq \ell_{(n)}$
2: Initialize current code to 0 (binary)
3: for $i = 1$ to $n$ do
4:     Assign current code to symbol with length $\ell_{(i)}$
5:     Increment current code (binary addition)
6:     Shift left by $\ell_{(i+1)} - \ell_{(i)}$ bits (if $i < n$)
7: end for
8: return The assigned codes
```

Example

Construct prefix code for lengths $\{2, 2, 3, 3\}$:

1. Sorted lengths: $\{2, 2, 3, 3\}$
2. Start with code = 00
3. Assign: symbol1 = 00, increment → 01
4. Assign: symbol2 = 01, increment → 10, no shift (same length)
5. Assign: symbol3 = 10, increment → 11, shift left 1 bit → 110
6. Assign: symbol4 = 110, done

Result: $\{00, 01, 10, 110\}$

0.22 Homework Assignment

1. For a source with probabilities $\{0.4, 0.3, 0.2, 0.1\}$:
    - Calculate the entropy

- Find optimal integer codeword lengths
- Verify they satisfy Kraft inequality
- Construct a prefix code with these lengths
- Calculate expected length and efficiency

2. Prove that for any prefix code with n codewords, the average length satisfies:

$$L \geq \log_2 n$$

   When does equality hold?

3. Consider a binary source with $P(0) = p$. Plot:
   - Entropy $H(p)$ vs p
   - Minimum achievable L vs p
   - The gap $L - H(p)$ for simple coding schemes

4. Research and explain one practical limitation of Shannon's theorem in real-world compression systems.

5. Design a prefix code for the alphabet $\{A, B, C, D, E, F\}$ with lengths $\{1, 2, 3, 3, 4, 4\}$. Show the binary tree representation.

0.23 Reading Assignment

- **Required**: Cover & Thomas, Chapter 5: Data Compression
- **Required**: Sayood, Chapter 2: Mathematical Preliminaries
- **Recommended**: MacKay, Chapter 5: Symbol Codes
- **Additional**: Original papers by Shannon (1948) and Kraft (1949)

0.24 Looking Ahead

In the next lecture, we will explore:

- Kolmogorov complexity: A different perspective on information
- Algorithmic information theory
- The concept of incompressibility
- Relationship between Kolmogorov complexity and practical compression